



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/676,743	09/30/2003	Anandhi Somasekaran	MSFT-2763/305222.1	7938
41505 7590 08/10/2007 WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION) CIRA CENTRE, 12TH FLOOR 2929 ARCH STREET PHILADELPHIA, PA 19104-2891			EXAMINER WANG, BEN C	
			ART UNIT 2192	PAPER NUMBER
			MAIL DATE 08/10/2007	DELIVERY MODE PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No. 10/676,743	Applicant(s) SOMASEKARAN ET AL.	
	Examiner Ben C. Wang	Art Unit 2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 30 September 2003.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-65 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-65 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|---|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date <u>11-19-2003</u> | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. Claims 1-65 are pending in this application and presented for examination.

Specification Objections

2. The specification is objected to because the following informalities:
 - "XLANG", e.g., cited in [0005], Line 2, is a registered trademark.
 - "Microsoft" and "IBM", e.g., cited in [0043], Line 6, are registered trademarks.
 - "BEA", e.g., cited in [0043], Line 12, is a registered trademark.

Appropriate correction is required (See MPEP § 608.01(b))

Claim Rejections – 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

3. Claims 46-59 and 60-65 are rejected under 35 U.S.C 101 because the claims are directed to non-statutory subject matter.
4. **As to claim 46**, the "computer-readable medium" is being cited, line 1, to include transmission media, light waves, a carrier wave etc., cited in [0029], lines 3-20 in the specifications; the claim is directed to a computer program product encoding a computer program. However, Applicant defines "computer-readable

medium" to include "a computer data signal embodied in a carrier wave". Signals and carrier waves do not fall within any class of statutory subject matter, and thus the claim is not limited to statutory subject matter. Please see Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility (1300 OG 142), Annex IV, Section (C) for details.

5. **As to claims 47-59**, they do not cure the deficiency of base claim 46, and also are rejected under 35 U.S.C. 101 as set forth above.

6. **As to claims 60-65**, they also are rejected under 35 U.S.C. 101 as set forth above claim 46.

Claim Rejections – 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claims 10-11, 24-25, 38-39, and 58-59 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Art Unit: 2192

8. **As to claims 10-11**, They have the same meaning in their respective claims; "where the instance state is being stored", cited in claim 10, and "where the instance state is being stored", cited in claim 11, these clauses have the same meaning.

9. **As to claims 24-25**, refer to above claims **10-11**, accordingly.

10. **As to claims 38-39**, refer to above claims **10-11**, accordingly.

11. **As to claims 58-59**, refer to claims **10-11**, accordingly.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

12. Claims 1-7 and 9-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ulrich Roxburgh, (*BizTalk Orchestration™: Transactions, Exceptions, and Debugging, Feb. 2001, Microsoft™*) (hereinafter 'Roxburgh') in view of Shaykat Chaudhuri, (*Debugging in Visual Studio .NET™, Jan. 2002, Microsoft™ Corporation*) (hereinafter 'Chaudhuri')

Art Unit: 2192

13. **As to claim 1**, Roxburgh discloses a business process service debugger for remotely debugging a business process service, comprising: means for reading stored state information regarding the business process service (e.g., Sec. of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another; Sec. of "Tracing Schedules", 2nd Par. – when these COM+ applications execute a schedule, they generate various events that can be trapped and displayed; BizTalk Server provides a tool, called the XLANG™ Event Monitor, to trap and display these events; The XLANG™ Event Monitor can subscriber to events published by host applications on any number of distributed computers, and can store these events for later analysis); and means for remotely debugging the business process service by way of the communications connection and according to the stored state information (e.g., Sec. of "Debugging Schedules" – to debug these executables, a combination of tracing and conventional debugging proves most effective; to debug the sequencing of a schedule (the flow from one action to another), tracing is useful; to debug the implementation of an individual action, traditional debugging mechanisms can be employed; by combining the two techniques, schedules can be debugged most effectively).

Roxburgh discloses the same way as debugging a standard COM+ component that is being called from a client application (e.g., Sec. of "Debugging Components in Schedules", 1st Par.), but does not explicitly disclose means for establishing a communications connection with a remote computer, wherein the remote computer is implementing the business process service.

Art Unit: 2192

However, in an analogous art of Debugging in Visual Studio .NET™, Chaudhuri discloses means for establishing a communications connection with a remote computer, wherein the remote computer is implementing the business process service (Sec. of "Debugging Multiple Processes Across Machines", 1st Par. – not only does the Visual Studio .NET™ debugger allow you to debug multiple processes at the same time, it also allows you to debug processes running on multiple machines simultaneously; with Visual Studio .NET™, you can launch multiple processes and debug them at the same time; 2nd Par. – alternatively, you can use the Process dialog box to specifically attach to processes on different machines and debug them simultaneous; remember that you need to have debugging components installed on the remote machine, and you need to have appropriate privileges to be able to debug the processes).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chaudhuri into the Roxburgh's system to further provide means for establishing a communications connection with a remote computer, wherein the remote computer is implementing the business process service in Roxburgh system.

The motivation is that it would further enhance the Roxburgh's system by taking, advancing and/or incorporating Chaudhuri's system which offers significant advantages for a single user interface for all the languages; the ability to debug multiple processes across multiple machines at the same time; powerful cross debugging between separate languages as once suggested by Chaudhuri (e.g., Sec. of "Introduction").

14. **As to claim 2** (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger further comprising means for displaying the business process service as a graphical image (e.g., the diagram on P. 8).

15. **As to claim 3** (incorporating the rejection in claim 2), Roxburgh discloses the business process service debugger wherein the graphical image comprises a workflow (e.g., the diagram on P. 8).

16. **As to claim 4** (incorporating the rejection in claim 2), Roxburgh discloses the business process service debugger further comprising means for interacting with the business process service according to a user instruction (e.g., Sec. of "Debugging Schedules").

17. **As to claim 5** (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein the stored state information corresponds to a variable assignment within the business process service (e.g., Sec. of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another).

18. **As to claim 6** (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein the stored state information is stored in a database (e.g., P. 2, 4th Par – Consistency – when committed, a

transaction must preserve the integrity of the data within the system; if a transaction performs a data modification in a database that was internally consistent before the transaction started, the database must still be internally consistent when the transaction is committed; 6th Par. – MSDTC (Microsoft™ Distributed Transaction Coordinator) was first released together with Microsoft™ SQL Server™ 6 and provides an object-based programming model for creating, destroying, managing, and monitoring transactions).

19. **As to claim 7** (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein the stored state information corresponds to message flow data (e.g., Sec. of “What Is a Transaction?”, 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another).

20. **As to claim 9** (incorporating the rejection in claim 1), Chaudhuri discloses the business process service debugger wherein said establishing means further comprises means for maintaining communications across a plurality of remote computers (Sec. of “Debugging Multiple Processes Across Machines”, 1st Par. – not only does the Visual Studio .NET™ debugger allow you to debug multiple processes at the same time, it also allows you to debug processes running on multiple machines simultaneously; with Visual Studio .NET™, you can launch multiple processes and debug them at the same time; 2nd Par. – alternatively, you can use the Process dialog box to specifically attach to processes on

different machines and debug them simultaneous; remember that you need to have debugging components installed on the remote machine, and you need to have appropriate privileges to be able to debug the processes).

21. **As to claim 10** (incorporating the rejection in claim 1) and **claim 11** (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein said debugging means comprises means for detecting a location where the instance state is being stored (e.g., P. 15, last Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor).

22. Claims 8, 12-39, and 46-59 are rejected under 35 U.S.C. 103(a) as being unpatentable over Roxburgh in view of Chaudhuri and further in view of Adams et al., (*BizTalk™ Unleashed 1st Edition, Feb. 2002, Sams Publishing*) (hereinafter 'Adams')

23. **As to claim 12**, Roxburgh discloses a system for remotely debugging a distributed transactional application, comprising: a server (e.g., Sec. of "Introduction", 5th Par. – this article discusses methodologies to debug and troubleshoot Biztalk Server Orchestration™ Services and BizTalk™ Messaging Services), wherein the server runs a business process service, thereby generating runtime data (e.g., Sec. of "What Is a Transaction?", 3rd Par. – a

transaction is an action or series of actions that transform a system from one consistent state to another).

Roxburgh discloses the same way as debugging a standard COM+ component that is being called from a client application (e.g., Sec. of "Debugging Components in Schedules", 1st Par.), but does not explicitly disclose a client computer for running a debugging user interface (UI) process, wherein the UI process establishes a communications connection with the server according to a user instruction, and further generates a runtime request pertaining to a location within the business process service.

However, in an analogous art of Debugging in Visual Studio .NET™, Chaudhuri discloses a client computer for running a debugging user interface (UI) process, wherein the UI process establishes a communications connection with the server according to a user instruction, and further generates a runtime request pertaining to a location within the business process service (Sec. of "Debugging Multiple Processes Across Machines", 1st Par. – not only does the Visual Studio .NET™ debugger allow you to debug multiple processes at the same time, it also allows you to debug processes running on multiple machines simultaneously; with Visual Studio .NET™, you can launch multiple processes and debug them at the same time; 2nd Par. – alternatively, you can use the Process dialog box to specifically attach to processes on different machines and debug them simultaneous; remember that you need to have debugging components installed on the remote machine, and you need to have appropriate privileges to be able to debug the processes).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chaudhuri into the Roxburgh's system to further provide for running a debugging user interface (UI) process, wherein the UI process establishes a communications connection with the server according to a user instruction, and further generates a runtime request pertaining to a location within the business process service in Roxburgh system.

The motivation is that it would further enhance the Roxburgh's system by taking, advancing and/or incorporating Chaudhuri's system which offers significant advantages for a single user interface for all the languages; the ability to debug multiple processes across multiple machines at the same time; powerful cross debugging between separate languages as once suggested by Chaudhuri (e.g., Sec. of "Introduction").

Roxburgh discloses a transaction is an action or series of actions that transform a system from one consistent state to another (e.g., Sec. of "What Is a Transaction?"), but Roxburgh and Chaudhuri do not disclose an interceptor for identifying the location within the business process service according to the runtime data and, when the location is identified, causing the server to carry out the runtime request.

However, in an analogous art of BizTalk™ Unleashed, Adams discloses an interceptor for identifying the location within the business process service according to the runtime data and, when the location is identified, causing the server to carry out the runtime request (e.g., Sec. of "Developing Custom

Tracking Solution” – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of “Example: A Custom Tracking Solution”, 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Adams into the Roxburgh-Chaudhuri’s system to further provide an interceptor for identifying the location within the business process service according to the runtime data and, when the location is identified, causing the server to carry out the runtime request. in Roxburgh-Chaudhuri system.

The motivation is that it would further enhance the Roxburgh-Chaudhuri’s system by taking, advancing and/or incorporating Adams’s system which offers significant advantages for building customized tracking solutions using the infrastructure provided by Biztalk server as once suggested by Adams (e.g., Sec. of “Developing Custom Tracking Solution”).

24. **As to claim 26**, Roxburgh discloses a method for debugging a business process service instance running on a remote computer, comprising: receiving a runtime request; and processing the runtime request at the remote computer with respect to within the instance (e.g., Sec. of “Introduction”, 5th Par. – this article discusses methodologies to debug and troubleshoot Biztalk Server

Orchestration™ Services and BizTalk™ Messaging Services; Sec. of “tracing Schedules”, 4th Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID)).

Roxburgh discloses the same way as debugging a standard COM+ component that is being called from a client application (e.g., Sec. of “Debugging Components in Schedules”, 1st Par.), but does not explicitly disclose if the business process service is in a debug mode, establishing a direct client connection channel with the remote computer.

However, in an analogous art of Debugging in Visual Studio .NET™, Chaudhuri discloses if the business process service is in a debug mode, establishing a direct client connection channel with the remote computer (Sec. of “Debugging Multiple Processes Across Machines”, 1st Par. – not only does the Visual Studio .NET™ debugger allow you to debug multiple processes at the same time, it also allows you to debug processes running on multiple machines simultaneously; with Visual Studio .NET™, you can launch multiple processes and debug them at the same time; 2nd Par. – alternatively, you can use the Process dialog box to specifically attach to processes on different machines and debug them simultaneous; remember that you need to have debugging components installed on the remote machine, and you need to have appropriate privileges to be able to debug the processes).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chaudhuri into the Roxburgh’s system to further provide if the business process service is in a

debug mode, establishing a direct client connection channel with the remote computer in Roxburgh system.

The motivation is that it would further enhance the Roxburgh's system by taking, advancing and/or incorporating Chaudhuri's system which offers significant advantages for a single user interface for all the languages; the ability to debug multiple processes across multiple machines at the same time; powerful cross debugging between separate languages as once suggested by Chaudhuri (e.g., Sec. of "Introduction").

Roxburgh discloses a transaction is an action or series of actions that transform a system from one consistent state to another (e.g., Sec. of "What Is a Transaction?"), but Roxburgh and Chaudhuri do not disclose causing an interceptor to monitor data regarding the business process service to find a location within the service based on stored state information.

However, in an analogous art of BizTalk™ Unleashed, Adams discloses causing an interceptor to monitor data regarding the business process service to find a location within the service based on stored state information (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Adams into the Roxburgh-Chaudhuri's system to further provide causing an interceptor to monitor data regarding the business process service to find a location within the service based on stored state information in Roxburgh-Chaudhuri system.

The motivation is that it would further enhance the Roxburgh-Chaudhuri's system by taking, advancing and/or incorporating Adams's system which offers significant advantages for building customized tracking solutions using the infrastructure provided by Biztalk server as once suggested by Adams (e.g., Sec. of "Developing Custom Tracking Solution").

25. **As to claim 46**, Roxburgh discloses a computer-readable medium having computer-executable instructions for performing a method for debugging a business process service instance running on a remote computer, comprising: receiving a runtime request; and processing the runtime request at the remote computer with respect to the instance (e.g., Sec. of "Introduction", 5th Par. – this article discusses methodologies to debug and troubleshoot Biztalk Server Orchestration™ Services and BizTalk™ Messaging Services; Sec. of "tracing Schedules", 4th Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID)).

Roxburgh discloses the same way as debugging a standard COM+ component that is being called from a client application (e.g., Sec. of "Debugging Components in Schedules", 1st Par.), but does not explicitly disclose if the

Art Unit: 2192

business process service is in a debug mode, establishing a direct client connection channel with the remote computer.

However, in an analogous art of Debugging in Visual Studio .NET™, Chaudhuri discloses if the business process service is in a debug mode, establishing a direct client connection channel with the remote computer (Sec. of "Debugging Multiple Processes Across Machines", 1st Par. – not only does the Visual Studio .NET™ debugger allow you to debug multiple processes at the same time, it also allows you to debug processes running on multiple machines simultaneously; with Visual Studio .NET™, you can launch multiple processes and debug them at the same time; 2nd Par. – alternatively, you can use the Process dialog box to specifically attach to processes on different machines and debug them simultaneous; remember that you need to have debugging components installed on the remote machine, and you need to have appropriate privileges to be able to debug the processes).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chaudhuri into the Roxburgh's system to further provide if the business process service is in a debug mode, establishing a direct client connection channel with the remote computer in Roxburgh system.

The motivation is that it would further enhance the Roxburgh's system by taking, advancing and/or incorporating Chaudhuri's system which offers significant advantages for a single user interface for all the languages; the ability to debug multiple processes across multiple machines at the same time; powerful

Art Unit: 2192

cross debugging between separate languages as once suggested by Chaudhuri (e.g., Sec. of "Introduction").

Roxburgh discloses a transaction is an action or series of actions that transform a system from one consistent state to another (e.g., Sec. of "What Is a Transaction?"), but Roxburgh and Chaudhuri do not disclose causing an interceptor to monitor data regarding the business process service to find a location within the service based on stored state configuration.

However, in an analogous art of BizTalk™ Unleashed, Adams discloses causing an interceptor to monitor data regarding the business process service to find a location within the service based on stored state configuration (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Adams into the Roxburgh-Chaudhuri's system to further provide causing an interceptor to monitor data regarding the business process service to find a location within the service based on stored state configuration in Roxburgh-Chaudhuri system.

The motivation is that it would further enhance the Roxburgh-Chaudhuri's system by taking, advancing and/or incorporating Adams's system which offers

Art Unit: 2192

significant advantages for building customized tracking solutions using the infrastructure provided by Biztalk server as once suggested by Adams (e.g., Sec. of "Developing Custom Tracking Solution").

26. **As to claim 8** (incorporating the rejection in claim 1), Roxburgh discloses a transaction is an action or series of actions that transform a system from one consistent state to another (e.g., Sec. of "What Is a Transaction?"), but Roxburgh and Chaudhuri do not disclose the business process service debugger wherein said reading means further comprises means for reading stored business process service configuration information.

However, in an analogous art of BizTalk™ Unleashed, Adams discloses the business process service debugger wherein said reading means further comprises means for reading stored business process service configuration information (e.g., Sec. of "Developing Custom Tracking Solutions" - the two primary tracking activities: configuring tracking settings and viewing tracking data).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Adams into the Roxburgh-Chaudhuri's system to further provide the business process service debugger wherein said reading means further comprises means for reading stored business process service configuration information in Roxburgh-Chaudhuri system.

The motivation is that it would further enhance the Roxburgh-Chaudhuri's system by taking, advancing and/or incorporating Adams's system which offers significant advantages for building customized tracking solutions using the infrastructure provided by Biztalk server as once suggested by Adams (e.g., Sec. of "Developing Custom Tracking Solution").

27. **As to claim 13** (incorporating the rejection in claim 12), Roxburgh discloses the system further comprising for storing business process service state information (e.g., Sec. of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another).

Adams discloses a database for receiving the runtime data (e.g., Sec. of "Developing Custom tracking Solutions" – we begin by covering the tracking database, including the core tables where interchange data, document data, and routing data is stored).

28. **As to claim 14** (incorporating the rejection in claim 13), Roxburgh discloses the system further comprising a display device, wherein the display device presents a shape corresponding to the business process service based on the business process service state information (e.g., P. 4, 2nd Par. – the transaction model for a schedule can be set by opening the properties dialog box for the Begin shape at the star of the schedule; Sec. of "What Is a Transaction?",

Art Unit: 2192

3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another).

Chaudhuri discloses an input device, wherein the input device receives the user instruction (Sec. of “Debugging Multiple Processes Across Machines”, 1st Par. – not only does the Visual Studio .NET™ debugger allow you to debug multiple processes at the same time, it also allows you to debug processes running on multiple machines simultaneously; with Visual Studio .NET™, you can launch multiple processes and debug them at the same time; 2nd Par. – alternatively, you can use the Process dialog box to specifically attach to processes on different machines and debug them simultaneous; remember that you need to have debugging components installed on the remote machine, and you need to have appropriate privileges to be able to debug the processes).

29. **As to claim 15** (incorporating the rejection in claim 14), Roxburgh discloses the system wherein the display device further presents a workflow representative of the program flow of the business process service (e.g., the diagram on P. 8).

30. **As to claim 16** (incorporating the rejection in claim 14), Roxburgh discloses the system wherein the display device further presents data representative of the message flow of the business process service (e.g., P. 5, the diagram; 1st Par. – the implementation of the schedule reads a message from

Art Unit: 2192

transactional message queue (receive queue) and write the message to another transaction message queue (send queue)).

31. **As to claim 17** (incorporating the rejection in claim 14), Roxburgh discloses the system wherein the shape is presented according to stored state information (e.g., Sec. of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another).

32. **As to claim 18** (incorporating the rejection in claim 12), Adams discloses the system wherein the database comprises a first database for receiving the runtime data and storing un-decoded tracking data (e.g., Sec. of "BizTalk™ Messaging" – BizTalk™ Messaging is a set of services that provide a mechanism for routing documents in an enterprise environment; Sec. of "Configuration Objects Versus BizTalk™ Messaging Objects" – Biztalk™ Messaging objects are the actual objects stored in the Biztalk™ Messaging database) and a second database for decoding and storing business process service tracking status information (e.g., Sec. of "Developing Custom tracking Solutions" – we begin by covering the tracking database, including the core tables where interchange data, document data, and routing data is stored).

33. **As to claim 19** (incorporating the rejection in claim 18), Adams discloses the system wherein the UI process comprises an application program interface

Art Unit: 2192

for communicating with the first database (e.g., Sec. of "BizTalk™ Messaging" – BizTalk™ Messaging is a set of services that provide a mechanism for routing documents in an enterprise environment; Sec. of "Configuration Objects Versus BizTalk™ Messaging Objects" – Biztalk™ Messaging objects are the actual objects stored in the Biztalk™ Messaging database).

34. **As to claim 20** (incorporating the rejection in claim 18), Adams discloses the system wherein the UI process comprises a UI component for communicating with the second database (e.g., Sec. of "Developing Custom tracking Solutions" – we begin by covering the tracking database, including the core tables where interchange data, document data, and routing data is stored).

35. **As to claim 21** (incorporating the rejection in claim 18), Adams discloses the system wherein the first database is a message box database (e.g., Sec. of "BizTalk™ Messaging" – BizTalk™ Messaging is a set of services that provide a mechanism for routing documents in an enterprise environment; Sec. of "Configuration Objects Versus BizTalk™ Messaging Objects" – Biztalk™ Messaging objects are the actual objects stored in the Biztalk™ Messaging database).

36. **As to claim 22** (incorporating the rejection in claim 18), Adams discloses the system wherein the second database is a tracking database (e.g., Sec. of "Developing Custom Tracking Solutions" – by covering the tracking database,

including the core tables where interchange data, document data, and routing data is stored; Sec. of "Maintaining the Tracking Database).

37. **As to claim 23** (incorporating the rejection in claim 12), Adams discloses the system wherein the interceptor is a component of a computer language that provides stored state tracking information (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window).

38. **As to claim 24** (incorporating the rejection in claim 12), Roxburgh discloses the system wherein the UI process detects a location where the instance is being processed (e.g., P. 15, last Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor).

39. **As to claim 25** (incorporating the rejection in claim 12), Roxburgh discloses the system wherein the UI process detects a location where the instance state is being stored (e.g., P. 15, last Par. – each instance in addition

Art Unit: 2192

has the unique identifier for the instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor).

40. **As to claim 27** (incorporating the rejection in claim 26), Adams discloses further comprising: querying a database containing a status of the business process service; displaying a query result on a display device; receiving user input with respect to the query result (e.g., Sec. of "Tracking Events in the Tracking Database", 1st Par. – the BizTalk™ Document Tracking Web application all you to query the database for document flows between organizations or applications; the XLANG™ events captured in the database can be viewed in relationship with the message flows displayed by document tracking Web application).

Chaudhuri discloses establishing the direct client connection channel in response to the user input (Sec. of "Debugging Multiple Processes Across Machines", 1st Par. – not only does the Visual Studio .NET™ debugger allow you to debug multiple processes at the same time, it also allows you to debug processes running on multiple machines simultaneously; with Visual Studio .NET™, you can launch multiple processes and debug them at the same time; 2nd Par. – alternatively, you can use the Process dialog box to specifically attach to processes on different machines and debug them simultaneous; remember that you need to have debugging components installed on the remote machine, and you need to have appropriate privileges to be able to debug the processes).

41. **As to claim 28** (incorporating the rejection in claim 27), Roxburgh discloses the information contained in the database is instance runtime data (e.g., Sec. of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another).

42. **As to claim 29** (incorporating the rejection in claim 27), Adams discloses the information contained in the database is instance tracking data (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window).

43. **As to claim 30** (incorporating the rejection in claim 26), Roxburgh discloses further comprising: creating the business process service using a process designer (e.g., Sec. of "Introduction", 1st Par. – with Microsoft BizTalk Orchestration Designer™, uses can design long-running business processes, specify an implementation for the individual actions that make up those processes, and compile this information into an executable XML representation, known as an XLANG™ schedule; the schedules created are distributed across time, organizations, and applications, in a loosely coupled and scalable manner).

Adams discloses saving a business process service configuration and symbolic data in a database; displaying a graphical representation of the business process service on a display device according to the saved business process service symbolic data (e.g., Sec. of “Developing Custom Tracking Solution” – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of “Example: A Custom Tracking Solution”, 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window).

Chaudhuri discloses generating a runtime request based on the graphical representation; and displaying a result of the runtime request on the display device (Sec. of “Debugging Multiple Processes Across Machines”, 1st Par. – not only does the Visual Studio .NET™ debugger allow you to debug multiple processes at the same time, it also allows you to debug processes running on multiple machines simultaneously; with Visual Studio .NET™, you can launch multiple processes and debug them at the same time; 2nd Par. – alternatively, you can use the Process dialog box to specifically attach to processes on different machines and debug them simultaneous; remember that you need to have debugging components installed on the remote machine, and you need to have appropriate privileges to be able to debug the processes).

Art Unit: 2192

44. **As to claim 31** (incorporating the rejection in claim 30), Roxburgh discloses the graphical representation comprises a shape corresponding to an operation in the business process service (e.g., the diagram on P. 8).

45. **As to claim 32** (incorporating the rejection in claim 30), Roxburgh discloses the graphical representation comprises a workflow representation of the business process service (e.g., the diagram on P. 8).

46. **As to claim 33** (incorporating the rejection in claim 30), Roxburgh discloses the saving step takes place in connection with compiling and deploying the business process service (e.g., Sec. of "Introduction", 1st Par. – with Microsoft BizTalk Orchestration Designer™, users can design long-running business processes, specify an implementation for the individual actions that make up those processes, and compile this information into an executable XML representation, known as an XLANG™ schedule; the schedules created are distributed across time, organizations, and applications, in a loosely coupled and scalable manner).

47. **As to claim 34** (incorporating the rejection in claim 30), Roxburgh discloses the business process service is implemented in a computer language that provides stored state information (e.g., Sec. of "Introduction", 1st Par. – with Microsoft BizTalk Orchestration Designer™, users can design long-running business processes, specify an implementation for the individual actions that

make up those processes, and compile this information into an executable XML representation, known as an XLANG™ schedule; the schedules created are distributed across time, organizations, and applications, in a loosely coupled and scalable manner).

48. **As to claim 35** (incorporating the rejection in claim 30), Chaudhuri discloses the interceptor request is a break point (e.g., Sec. of “Web Service Debugging”, 1st Par. – both web applications and web services can be easily debugged with Visual Studio .NET; Just set your breakpoints in you web service code and invoke the method through the test page; Sec. of “No More Additional DLLs”).

49. **As to claim 36** (incorporating the rejection in claim 30), Roxburgh discloses the runtime request is a request for data regarding an instance of the business process service (e.g., Sec. of “Introduction”, 5th Par. – this article discusses methodologies to debug and troubleshoot Biztalk Server Orchestration™ Services and BizTalk™ Messaging Services; Sec. of “tracing Schedules”, 4th Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID)).

50. **As to claim 37** (incorporating the rejection in claim 36), Roxburgh discloses the data regarding the instance is state information (e.g., Sec. of “What

Art Unit: 2192

Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another).

51. **As to claim 38** (incorporating the rejection in claim 26), Roxburgh discloses further comprising detecting a location where the instance is being processed (e.g., P. 15, last Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor).

52. **As to claim 39** (incorporating the rejection in claim 26), Roxburgh discloses further comprising detecting a location where an instance state is being stored (e.g., P. 15, last Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor).

53. **As to claim 47-59**, refer to above **claim 27-39**, accordingly.

Claim Rejections – 35 USC § 102(b)

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102(b) that form the basis for the rejections under this section made in this office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

54. Claims 40-44, and 60-64 are rejected under 35 U.S.C. 102(b) as being anticipated by Adams.

55. **As to claim 40**, Adams discloses a method in a computer system for displaying on a display device a business process service debugger, the method comprising: querying a database for tracking information regarding the business process service, wherein the tracking information contains an operation identifier; receiving a query result and generating a shape representative of the operation according to the identifier; and presenting the shape, the tracking information according to the query result, and a debugging option on the display device (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window).

56. **As to claim 60**, Adams discloses a computer-readable medium having computer-executable instructions for performing a method for displaying on a display device a business process service debugger, the method comprising: querying a database for configuration information regarding the business process service wherein the configuration information contains an operation identifier; receiving a query result and generating a shape representative of the operation

Art Unit: 2192

according to the identifier; and presenting the shape, the configuration information according to the query result, and a debugging option on the display device (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window).

57. **As to claim 41** (incorporating the rejection in claim 40), Adams discloses further comprising receiving runtime data for the business process service and presenting the runtime data on the display device (e.g., Sec. of "Biztalk Messaging").

58. **As to claim 42** (incorporating the rejection in claim 41), Adams discloses wherein the runtime data comprises message flow information (e.g., Chapter 10 – Using the BizTalk Orchestration Designer, 2nd Par. – the BizTalk Orchestration Designer contains wizards that guide you through creating and configuring ports and message flows).

59. **As to claim 43** (incorporating the rejection in claim 40), Adams discloses wherein the runtime data comprises message flow information (e.g., Chapter 10 – Using the BizTalk Orchestration Designer, 2nd Par. – the BizTalk Orchestration

Art Unit: 2192

Designer contains wizards that guide you through creating and configuring ports and message flows).

60. **As to claim 44** (incorporating the rejection in claim 40), Adams discloses further comprising presenting a content of a message according to the process (e.g., Sec. of "Defining Data Flow Through Message", 1st Par. – a message definition is created when Action shape is connected to a port; Sec. of "XML Communication Wizard", 2nd Par.).

61. **As to claim 41-44**, refer to above **claim 61-64**, accordingly.

62. Claims 45 and 65 are rejected under 35 U.S.C. 103(a) as being unpatentable over Adams in view of Chaudhuri.

63. **As to claim 45** (incorporating the rejection in claim 40) and **claim 65** (incorporating the rejection in claim 60), Adams does not disclose further comprising receiving input from an input device to place a break point proximate a shape, and presenting a symbol representing the break point on the display device.

However, in an analogous art of Debugging in Visual Studio .NET™, Chaudhuri discloses further comprising receiving input from an input device to place a break point proximate a shape, and presenting a symbol representing the break point on the display device.

Art Unit: 2192

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chaudhuri into the Roxburgh's system to further provide means for establishing a communications connection with a remote computer, wherein the remote computer is implementing the business process service in Roxburgh system.

The motivation is that it would further enhance the Roxburgh's system by taking, advancing and/or incorporating Chaudhuri's system which offers significant advantages for a single user interface for all the languages; the ability to debug multiple processes across multiple machines at the same time; powerful cross debugging between separate languages as once suggested by Chaudhuri (e.g., Sec. of "Introduction").

Conclusion

64. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.


Art Unit: 2192

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

BCW

BW

July 26, 2007



TUANDAM
SUPERVISOR PATENT EXAMINER